Model Based (Industrial and) Systems Engineering

Leon F. McGinnis
Gwaltney Professor of Manufacturing Systems
Georgia Institute of Technology
April 27, 2011
Model Based Industrial Engineering

Contemporary IE education certainly could be said to be “model centric”, given our emphasis on mathematics, optimization, statistics, and simulation. Yet compared to software engineers, we are relatively immature users and creators of models, modelling languages, and modelling tools. This talk will describe the opportunity that I think we are missing, how we might capture that opportunity, and what doing so might mean for our profession. I will give some examples of what can be accomplished when we first focus our modelling work on describing the thing we are engineering, in its own terms, and then use that model as a platform for accessing the wide variety of analysis and synthesis tools already at our disposal.
Agenda

- Engineering with Models
- Model Driven Architecture
- Software-to-IE Analogy
- MDA and Factory Simulation
- R&D Opportunities
Caveats

- Perspective = Discrete Event Logistics Systems
- Overly simplistic and superficial treatment of MDA
Mechanical Engineering Models
Aerospace Engineering Models
Electrical Engineering Models
IE Example
Engineering IS Modeling

Artifact of Interest → Direct Abstraction → Analysis Abstraction

Domain of Interest

Interpretation

Analysis Result

Analysis Tool
ME, AE, EE Domains

Artifact of Interest

Domain of Interest

Direct Abstraction

Interpretation

Analysis Abstraction

Analysis Result

Analysis Tool
What if I need more than one analysis to support artifact design?
What am I really talking about?
What is a model?

- Abstract representation of something
- Articulated in a specialized “language”
How do we teach warehousing?

- Order Pick Models
- Sortation Models
- Storage Models
- Flow Models
- Layout Models
What if?

- Layout Models
- Flow Models
- Order Pick Models
- Sortation Models
- Storage Models
This would require…

- Formal, graphical language for warehousing and for warehousing analysis and design
- Authoring tools for creating direct warehouse models
- Integrating warehouse analysis models with the direct warehouse model
Usual excuses…

- Problems in our domain are too complex, too multi-faceted
  - Really? Compared to airplanes or microprocessors or large software systems?
- It’s not an intellectually challenging problem—it’s just clever programming
  - Really? So are VHDL and NX are just clever programming?
- It’s not our job…
  - Really? Then who should be doing it?
A quick and superficial overview of “model driven engineering” from the software engineering domain.
Legacy Document-Based Approach

Business Analyst → Requirements Document
Architect → Architectural Design Document
Designer → Detailed Design Document
Developer → Code
Model Driven Architecture Approach

Business Analyst

Architect

Designer

Developer

Computation Independent Model

Platform Independent Model

Platform Specific Model

Code

Model Transformation Technology

Turn implementation into a computational process, i.e., make it algorithmic.
Essential MDA Technologies

MOF
Meta Object Facility

QVT
Query/View/Transformation

UML
Unified Modeling Language

XMI/XML
eXtensible Markup Language

DSL
Domain Specific Language

MDA
Model Driven Architecture
A model transformation in *Model Driven Engineering* takes as input a model conforming to a given meta model and produces as output another model conforming to another given meta model.
Transformation in the MOF Framework

- MOF/ECORE
- UML MetaModel
- QVT MetaModel
- Java MetaModel
- UML-Java Mapping
- UML Model
- Xform Engine
- Java “Model”
We can appropriate this technology to our own purposes in modeling and analyzing DELS!
Turn implementation into a computational process, i.e., make it algorithmic.
One more technology piece …
Systems Modeling Language: SysML

- SysML is an extension of the Unified Modeling Language.
- Supports the modeling of physical systems – not just software.
- It is not a methodology or tool.
- Improves the ability to exchange systems engineering information among tools and people.
Pillars of SysML: 4 Main Diagram Types

1. Structure

```
+-----------------+                        +-----------------+
| Library::       | Library::Electro-Hydraulic     |
| Electronic      | Valve                       |
| Processor       |                            |
+-----------------+                        +-----------------+ |
|                 | Library::Anti-Lock Controller |                 |
|                 |                            |
+-----------------+                        +-----------------+ |
| Traction        | Brake Modulator             |
| Detector        |                            |
```

2. Behavior

3. Parametrics

4. Requirements

(Source: Friedenthal, www.omgsysml.org)
Pillars of SysML: 4 Main Diagram Types

1. Structure

```
becca [package] VehicleStructure [ABS-Block Definition Diagram]
```

- «block» Library::Electronic Processor
- «block» Anti-Lock Controller
- «block» Traction Detector
- «block» Library::Electro-Hydraulic Valve

```
ibd [block] Anti-LockController [Internal Block Diagram]
```

- d1:Traction Detector
- m1:Brake Modulator

```
c1:modulator interface
```

2. Behavior

```
interaction
```

```
sd ABS_ActivationSequence [Sequence Diagram]
```

- d1:Traction Detector
- m1:Brake Modulator

```
detTrkLos()
sendSignal()
modBrkFrc(traction_signal:boolean)
```

```
modBrkFrc()
SendAck()
```

(Source: Friedenthal, www.omgsysml.org)
Pillars of SysML: 4 Main Diagram Types

1. Structure

- bdd [package] VehicleStructure [ABS-Block Definition Diagram]
  - «block» Library::Electronic Processor
  - «block» Anti-Lock Controller
  - «block» Library::Electro-Hydraulic Valve
  - d1:Traction Detector
  - ibd [block] Anti-LockController [Internal Block Diagram]
  - c1:modulator interface
  - m1:Brake Modulator

2. Behavior

- interaction
- state
- machine

- stm TireTraction [State Diagram]
  - Gripping
  - Slipping
  - LossOfTraction
  - RegainTraction
  - modBrkFrc(traction_signal:boolean)
  - modBrkFrc()
  - SendAck()

3. Parametrics

4. Requirements
Pillars of SysML: 4 Main Diagram Types

1. Structure

2. Behavior

3. Parametrics

4. Requirements

(Source: Friedenthal, www.omg.sysml.org)
Proof of Concept Demonstration

In collaboration with Rockwell Collins, Cedar Rapids, IA
Analyzing avionics assembly process

- Quickly and cost effectively generate simulation models of new products/programs
- Consider the BOM, the available resources, any scheduling constraints
- What is the sustainable throughput?
- How much WIP will be required?
- What is the manufacturing lead time?
Legacy approach

Custom built
Arena simulator
Pros and Cons

- **Pros**
  - Puts the analysis in the hands of the factory engineers

- **Cons**
  - Have to maintain both the Excel front-end and the simulator, and keep them “synchronized”
  - Not effective for communicating changes between factory engineers and simulationists
  - Not very useful to other potential stakeholders
Proof of Concept Demonstration

- Create “domain specific language” – DSL
- Model process using DSL
- Transform resulting process model to Arena using ATL
- Reproduce simulation results from existing simulator
Proof of Concept for Simulation

Representative Manufacturing Systems DSL in SysML

Conforms to

Manufacturing SysML model

ATL Transformation Script

Use

Model Transformation

Arena RDB Meta Model

Conforms to

Arena Simulation Model in RDB

Execute

Arena Software
Example System: assembly and integration processes for electronic components to produce module in a manufacturing system consisting of:
- 13 electronic components,
- 20 human resources, and
- 14 machine resources.

Each electronic component goes through steps of manufacturing process and inspection/testing that require human resource(s) and/or machine resource(s). The testing/inspection involves a yield rate that represents a probability of passing the test.
Profile: based on SysML, a domain specific language (DSL) is developed to build models using the common language used in manufacturing facilities, i.e. operator, machine, part, etc. The DSL is implemented as a “profile”
### Process Plan

<table>
<thead>
<tr>
<th><strong>SysML in MagicDraw / 2</strong></th>
</tr>
</thead>
</table>

**Process Plan**

#### «stereotype» ProcessPlan [Activity]

**Manufacturing Activities: Offline, Build, Inspection, Test, and ESS**

<table>
<thead>
<tr>
<th><strong>OfflineProcess [CallAction]</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>+ batchSize : String = 1</td>
</tr>
<tr>
<td>+ processTime : String</td>
</tr>
<tr>
<td>+ transferTime : String = 0</td>
</tr>
<tr>
<td>+ moveResources : Human_Resources [1..*]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>BuildProcess [CallAction]</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>+ batchSize : String = 1</td>
</tr>
<tr>
<td>+ processTime : String</td>
</tr>
<tr>
<td>+ humanResources : Human_Resources [1..*]</td>
</tr>
<tr>
<td>+ transferTime : String = 0</td>
</tr>
<tr>
<td>+ moveResources : Human_Resources [1..*]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TestProcess [CallAction]</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>+ batchSize : String = 1</td>
</tr>
<tr>
<td>+ isAutomated : Boolean</td>
</tr>
<tr>
<td>+ setupTime : String</td>
</tr>
<tr>
<td>+ processTime : String</td>
</tr>
<tr>
<td>+ teardownTime : String</td>
</tr>
<tr>
<td>+ moveResources : Human_Resources [1..*]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ESSManufacturingActivity [CallAction]</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>+ humanResources : Human_Resources [1..*]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>InspectionProcess [CallAction]</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>+ batchSize : String = 1</td>
</tr>
<tr>
<td>+ yieldRate : String</td>
</tr>
<tr>
<td>+ processTime : String</td>
</tr>
<tr>
<td>+ transferTime : String</td>
</tr>
<tr>
<td>+ moveResources : Human_Resources [1..*]</td>
</tr>
</tbody>
</table>

**Process plan is comprised of manufacturing activities**
Resources: Operators, Movers, and Machines

- **Human Resources**
  - **HumanResource** (Class)
    - capacity: String = 1
    - schedule: Shift [1]
  - **GroupOfHumanResources** (Class)
    - groupElements: HumanResource [1..*]

- **Mover Resources**
  - **MoverResource** (Class)
    - capacity: String = 1
    - schedule: Shift
  - **GroupOfMoverResources** (Class)
    - groupElements: MoverResource [1..*]

- **Machine Resources**
  - **MachineResource** (Class)
    - capacity: String = 1
  - **GroupOfMachineResources** (Class)
    - groupElements: MachineResource [1..*]

Resources work in shifts
**Containment Tree:** Containment tree has to be structured in a certain order.

**Package 1:** Production Schedules

**Package 2:** Working Shifts
Package 3: Manufacturing Part owns a Process Plan which is a collection of Manufacturing Activities.
Package 4: Resources and group of resources
Bill of Material

Stereotyped from DSL
Process Plan:

<OfflineProcess>
  ReceiveBackplane
  {batchSize = "1", moverResources = Mover1, processTime = "180", transferTime = "10")
</OfflineProcess>

<BuildProcess>
  BuildBackplane
  {batchSize = "1", dryTime = "0", humanResources = OperatorBuildGroup, OperatorInspectionGroup, processTime = "Uniform(40,60)")
</BuildProcess>

<TestProcess>
  TestBackplane
  {batchSize = "2", humanResources = TechnicianTestGroup, isAutomated, machineResources = MachineTestGroup, moverResources = Mover1, processTime = "5", reworkHumanResources = Operator3, reworkTime = "6", setupTime = "2", teardownTime = "4", transferTime = "4", troubleshootHumanResources = OperatorBuildGroup, troubleshootMachineResources = MachineTestGroup, troubleshootTime = "2", yieldRate = "94")
</TestProcess>

<InspectionProcess>
  InspectBackplane
  {batchSize = "3", humanResources = Operator1, processTime = "10", reworkHumanResources = OperatorInspectionGroup, reworkTime = "5", yieldRate = "91")
</InspectionProcess>
From SysML to XMI

- SysML has an Export option to **Eclipse UML2 XMI File**
ATL Configuration transforms models based on their metamodels:

- Metamodels conform to Ecore/MOF
- SysML IN as a UML File
- Arena OUT as a database (xml)

XMI to XML
- Access database file is imported into Arena as:
What is the advantage of this approach?

- Adding new elements of already defined “features” requires only a change to the modeling library; transformation is unchanged => much easier to maintain.
- Enables common practice across multiple sites, products, etc.
- New “features” can be discussed using existing graphical model as basis.
- Transformation model is relatively easy to work with in adding new “features”; modularity.
- Change to new solver => change only transformation, not factory models.
- Platform for additional stakeholders’ analyses, e.g., capacity planning and capital budgeting.
Other similar demonstrations

- Global supply chain for wind turbine manufacturer—optimization, financial analysis
- Functional design of supply chain management organization—organization design
- Integrated systems design of conventional warehouse—specification, including control, performance evaluation, simulation
What if?

- Layout Models
- Flow Models
- Order Pick Models
- Sortation Models
- Storage Models
Research opportunities

- Domain specific languages
- Analysis meta-models
- Modular, reusable transformation components
- Rethinking system modeling: structure, behavior, function, decisions
- Integration to create prototype engineering tools:
  - Semantic integration
  - Workflow integration
Could this change our discipline?

- New modeling skill for our arsenal
- Rationalize the modeling process
  - Domain & application modeling
  - Analysis & decision support modeling
- “Off line” modeling vs. “on line” modeling
- Multi-stakeholder modeling
  - Which is not the same as multi-criterion modeling
Questions?
Comments?